What is claimed is:

Sub-a1

1. An application programming interface (API) for enhancing data network communication, comprising:

an identify address function including programming instructions for identifying a stored node address stored by a base driver for a network interface associated with the base driver; and

an update node address function including programming instructions for directing the base driver to update the stored node address with a new node address in a configuration storage of the base driver, and in a receive address filtering table for the network interface.

2. The API of claim 1, wherein the identify address function includes submitting a request to the base driver, to which is received a response including the node address stored by the base driver.

3. The API of claim 1, wherein the identify address function includes programming instructions for inspecting the configuration storage of the base driver, such storage having an entry identifying the stored node address.

4. An API according to claim 1, further comprising:

a driver identification function including programming instructions for sending an identity-check request to the base driver, said driver providing a response selected from a group consisting of: a predetermined identifier, a base driver revision number, and an identification of a vendor of the base driver.

5. An API according to claim 4, wherein the predetermined identifier is a copyright string for the vendor of the base driver.

6. An article of manufacture, comprising a computer readable medium having encoded thereon programming instructions capable of directing a processor to perform operations of:

an identify address function for identifying a stored node address stored by a base driver for a network interface associated with the base driver; and

an update node address function for directing the base driver to update the stored node address with a new node address in a configuration storage of the base driver, and in a receive address filtering table for the network interface.

7. An API according to claim 1, further comprising:

a first transmission function including programming instructions for re-transmitting data, received in a compatible format from a network source, in an incompatible format to a network destination; and

a second transmission function including programming instructions for re-transmitting data, received in the incompatible format from the network destination, in the compatible format to the network source.

8. An API according to claim 7, further comprising:

a report capabilities function including programming instructions for sending the base driver a request to have the base driver report its capabilities;

a receive capabilities function including programming instructions for receiving a response including said capabilities;

wherein the incompatible format is formatted according to said capabilities.

9. An API according to claim 7, further comprising:

a virtual LAN function including programming instructions to direct the base driver to enter a desired virtual LAN operative state; and

a disconnect function including programming instructions to notify the base driver that the API has concluded communications with the base driver.

10. An article of manufacture, comprising a computer readable medium having encoded thereon instructions to direct a processor to perform operations of:

a first transmission function for re-transmitting data, received in a compatible format from a network source, in an incompatible format to a network destination; and a second transmission function for re-transmitting data, received in the incompatible format from the network destination, in the compatible format to the network source.

11. An API according to claim 1 for providing transparent fail-over from a first network interface to a second network interface, further comprising:

a status function including programming instructions for polling a first base driver for the first network interface to detect a failure of said first network interface;

wherein the update node address function includes a function to direct a second base driver for the second network interface to store the node address of the first network interface as the stored node address for the second base driver.

12. An API according to claim 11, in which a Novell ODI compliant network is utilized for network communication, and wherein the update node address function uses at least one ODI MLID Control Routine.

13. An article of manufacture, comprising a computer readable medium having encoded thereon instructions to direct a processor to perform an API having:

an identify address function for identifying a stored node address stored by a base driver for a network interface associated with the base driver;

an update node address function for directing the base driver to update the stored node address with a new node address;

a status function in communication with a first base driver for the first network interface to detect a failure of the first network interface; and

a function to direct a second base driver for the second network interface to store the node address of the first network interface as the stored node address for the second base driver.

14. An API according to claim 1 for providing transparent load balancing of data transmissions directed towards the network interface by distributing such data across a second network interface, further comprising:

a queue monitoring function including programming instructions for detecting a workload of the first network interface; and

a distribution function including programming instructions for routing a portion of said data transmissions through the second network interface, said distribution function utilizing the update node address function to associate the node identifier of the first network interface with the second network interface.

15. A networking method, comprising:

receiving first network traffic with a protocol stack;

sending said first traffic to an intermediary layer;

routing said first traffic to a virtual interface driver;

repackaging said first traffic by the virtual interface driver, and providing said repackaged traffic to a virtual protocol stack;

sending said repackaged traffic to the intermediary layer; and

routing said repackaged traffic by the intermediary layer to an interface driver for the network interface.

16. A method according to claim 15, further comprising:

locating a fail over network interface having a node address memory;

identifying a failed network interface having a node address;

storing the node address in the node address memory; and

routing network traffic for the failed network interface through the fail over network interface.

17. A method according to claim 16, further comprising:

wherein said first network traffic is received in a first protocol format, and said repackaged traffic is in a second network protocol format different from the first protocol format.

18. A method according to claim 16, wherein locating the fail over network interface comprises:

submitting an node identification request to a base driver for a potential fail over network interface; and

receiving a response from said driver, said response including an authentication string;

verifying said authentication string has a predetermined value before said potential fail over network interface is used as the fail over network interface.

19. An article of manufacture, comprising a computer readable medium having encoded thereon instructions to direct a processor to perform the operations of:

receiving first network traffic with a protocol stack;

sending said first traffic to an intermediary layer;

routing said first traffic to a virtual interface driver;

repackaging said first traffic by the virtual interface driver, and providing said repackaged traffic to a virtual protocol stack;

sending said repackaged traffic to the intermediary layer; and

routing said repackaged traffic by the intermediary layer to an interface driver for the network interface.

20. A method for redundant networking in a network environment, comprising:

determining operative status of a first network interface having a first driver, and of a second network interface having a second driver with a driver memory for storing a MAC address for said second interface;

if the first network interface is inoperative, instructing the second driver to store the first network interface MAC address in the driver memory to allow processing by the second network interface of network traffic bound for the first network interface;

directing the second driver to activate the second network interface; and

directing the first driver to deactivate the first network interface.

21. A method according to claim 20, in which the network environment is a Novell based network, and wherein ODI commands are issued to said first and second drivers.

22. A method according to claim 21, further comprising:

receiving first network traffic by a protocol stack;

forwarding said first network traffic to a LSL;

routing said first network traffic from the LSL to a virtual MLID, and deriving second network traffic from said first network traffic;

providing said second network traffic to a virtual protocol stack; and

forwarding said second network traffic to the LSL.

23. A method for load balancing transmissions over a network, comprising:

providing a first network interface having a first driver, said first interface having a communication load;

providing a second network interface having a second driver with an updateable driver memory for storing a MAC address;

receiving data from a network source;

apportioning said received data into a first and a second portion;

instructing the second driver to store the first network interface MAC address in the updateable driver memory;

sending the first portion to the first driver for network transmission; and

sending the second portion to the second driver for network transmission.

24. A method according to claim 23, further comprising:

wherein a communication protocol is bound to a virtual network interface for receiving the data from the network source, and wherein said apportioned data is re-transmitted by a virtual protocol stack directing said apportioned data to the first and second drivers.

25. A method according to claim 24, wherein the virtual protocol stack repackages said apportioned data in a format unsupported by the network source.

26. A system, comprising:

~~means for identifying a stored node address stored by a base driver for a network interface associated with the base driver; and~~

~~means for directing the base driver to update the stored node address with a new node address.~~

~~27. A system according to claim 26, further comprising:~~

~~means for re-transmitting data, received in a first format from a network source, in a second format to a network destination; and~~

~~means for re-transmitting data, received in the second format from the network destination, in the first format to the network source.~~

add B'